LISTING OF CLAIMS:

1. (Previously presented) A method to execute an instruction on an operand stack, the method comprising:

performing a stack-state-aware translation of the instruction to threaded code to determine an operand stack state for the instruction, including determining an entry point into shared execution code based on the stack state;

dispatching the instruction according to the operand stack state for the instruction; and

executing the instruction.

2. (Original) The method according to claim 1, said performing comprising:

determining a number of operands on the operand stack before the instruction is executed;

determining a number of operands on the operand stack after the instruction is executed based on a number of operands that the instruction consumes and a number of operands that the instruction produces; and

inferring a number of shift operations required after execution of the instruction to maintain top-of-stack elements.

3. (Original) The method according to claim 2, wherein the number of shift operations required after execution of the instruction is based on the number of operands on the operand

stack before the instruction is executed and the number of operands on the operand stack after the instruction is executed.

4. (Original) The method according to claim 2, wherein the number of shift operations required after execution of the instruction is inferred based on a static lookup table.

5. (Original) The method according to claim 1, wherein the operand stack is a mixed-register stack.

6. (Original) The method according to claim 1, wherein the operand stack state comprises a number of shift operations to maintain top-of-stack elements of the operand stack after the execution of the instruction.

7. (Original) The method according to claim 6, wherein the top-of-stack elements comprise a register stack.

8. (Original) The method according to claim 1, further comprising:

refilling the operand stack.

9. (Previously presented) A system comprising:

an operand stack to execute an instruction; and

an interpreter to determine a state of the operand stack, translate the instruction into threaded code, and dispatch the instruction based on the state of the operand stack, wherein said interpreter is further to determine an entry point into shared execution code based on the stack state.

10. (Original) The system according to claim 9, wherein the operand stack is a mixed stack comprising a register stack and a memory stack.

11. (Original) The system according to claim 10, wherein the register stack comprises at least one register to hold at least one respective top element of the stack and the memory stack comprises a contiguous memory region to hold the remaining elements of the operand stack.

12. (Previously presented) A machine accessible medium containing program instructions that, when executed by a processor, cause the processor to perform a series of operations comprising:

translating a virtual machine instruction into threaded code based on an operand stack state of the virtual machine instruction, including determining an entry point into shared execution code based on the stack state;

dispatching the virtual machine instruction according to the operand stack state; and

executing the instruction.

13. (Original) The machine accessible medium according to claim 12, wherein the threaded code is based on an entry point into shared execution code.

14. (Original) The machine accessible medium according to claim 12, further containing program instructions that, when executed by the processor cause the processor to perform further operations comprising:

determining a number of operands that are present on an operand stack at a time before the virtual machine instruction is executed;

determining a number of operands that are present on the operand stack at a time after the virtual machine instruction is executed; and

inferring a number of shift operations required to maintain top-of-stack elements after the virtual machine instruction is executed.

15. (Original) The machine accessible medium according to claim 13, wherein the wherein the number of shift operations required after execution of the instruction is based on the number of operands present on the operand stack at a time before the instruction is executed and the number of operands present on the operand stack at a time after the instruction is executed.

16. (Original) The machine accessible medium according to claim 13, wherein the number of shift operations required after execution of the instruction is inferred based on a static lookup table.

17. (Original) The machine accessible medium according to claim 12, wherein the operand stack state comprises a number of shift operations to maintain top-of-stack elements of an operand stack after execution of the virtual machine instruction.

18. (Original) The machine accessible medium according to claim 17, wherein the top-of-stack elements comprise a register stack.

19. (Original) The machine accessible medium according to claim 12, further containing program instructions that, when executed by the processor cause the processor to perform further operations comprising:

execute a number of shift operations to replace top-of-stack elements to an operand stack.

20. (Original) The machine accessible medium according to claim 19, wherein the number of shift operations is based on a number of elements on the operand stack that are consumed by the virtual machine instruction and a number of elements that are produced by the virtual machine instruction.

21.-23. (Cancelled)